5th International Conference on Emerging Trends in Engineering, Technology, Science and Management
Institution of Electronics and Telecommunication Engineers, Ganganagar, Bengaluru, Karnataka, India (ICETETSM-17)
20th August 2017, www.conferenceworld.in ISBN: 978-93-86171-60-3

An Ameliorated Methodology to Eliminate Redundancy in Databases Using SQL

Praveena M V[1], Dr. Ajeet A. Chikkamannur[2]

[1]Department of CSE, Dr Ambedkar Institute of Technology, VTU, Karnataka, India

[2]Department of CSE, R L Jalappa Institute of Technology, VTU, Karnataka, India

ABSTRACT

Today, there is an exponential growth in the amount and variety of data stored in the form of databases. Data Cleaning is an inevitable problem when combining data from various distributed and commercial databases. Duplication elimination is an integral part of data integration and cleaning. While many frameworks and approaches exist to identify, merge and eliminate duplicates in the databases that relies on ad hoc solutions.

We propose a complementary approach to eliminate redundancy in databases using Structured Query Language (SQL). SQL is the fourth generation database language and has statements for data definition, query and update. Today, most relational and commercial databases support SQL. This work is done by using SQL statements to eliminate redundant data in the relational databases. This new methodology can be easily embedded into any application programs which support database approaches.

Keywords – Distinct, Dynamic SQL, Integrity, Normalization, Redundancy, Static SQL

I. INTRODUCTION

The main objective of designing any good relational databases is to define a set of relational schemas without data redundancy and data anomalies. The major problematic areas in the good database design are repetition of information and inability to represent correct information. The repetition of information in databases is undesirable; consume space and leads to complexity during modification. When designing relational database schema, we need to eliminate redundancy of data without losing any data. Data redundancy means data is repeated in different rows of a table or indifferent tables in the database. If any data found in two different locations in database (direct redundancy) or if data can be calculated from other data items using SQL query processing (indirect redundancy), then the data is said to contain redundancy.

In the interests of convenience and performance, it may be acceptable to tolerate data redundancy in databases. Consider a classic widespread example – the use of PIN CODE within an address. Generally address consisting of a street, city and state then inclusion of a PIN CODE is actually redundant. This also breaks normalization process because the address is functionally dependent on the PIN CODE. That means given a PIN CODE, the city and street is predetermined. The alternative approach here requires a separate table mapping the PIN CODE to the city and street address. Normally this approach is not implemented since the problems created by the redundancy are far outweighed by the convenience [1].

Normalization is the standard process of taking data from a real world problem and reducing it to a set of relations while ensuring data integrity and eliminating redundancy in data. It avoids unnecessary duplication of data and improves the logical design, and also promotes integrity.

In this paper, we present a new improved methodology to eliminate redundancy in relational databases using SQL. Duplicate rows can appear more than once in a database table, also in the result of a query processing. But SQL does not automatically eliminate redundant rows resulted from the query processing. It is because eliminating redundancy is an expensive process and sometimes user needs duplicate rows in the result of a query [2]. Hence we use distinct keyword in SQL to eliminate redundancy. These results only distinct rows should remain in the resultant table. Both Static SQL and Dynamic SQL statements are used to eliminate redundancy in relational, distributed and commercial databases.

II. STATIC AND DYNAMIC SQL

To process an SQL statement the RDBMS (Relational Database Management System) follows the steps shown in Figure 1.1.

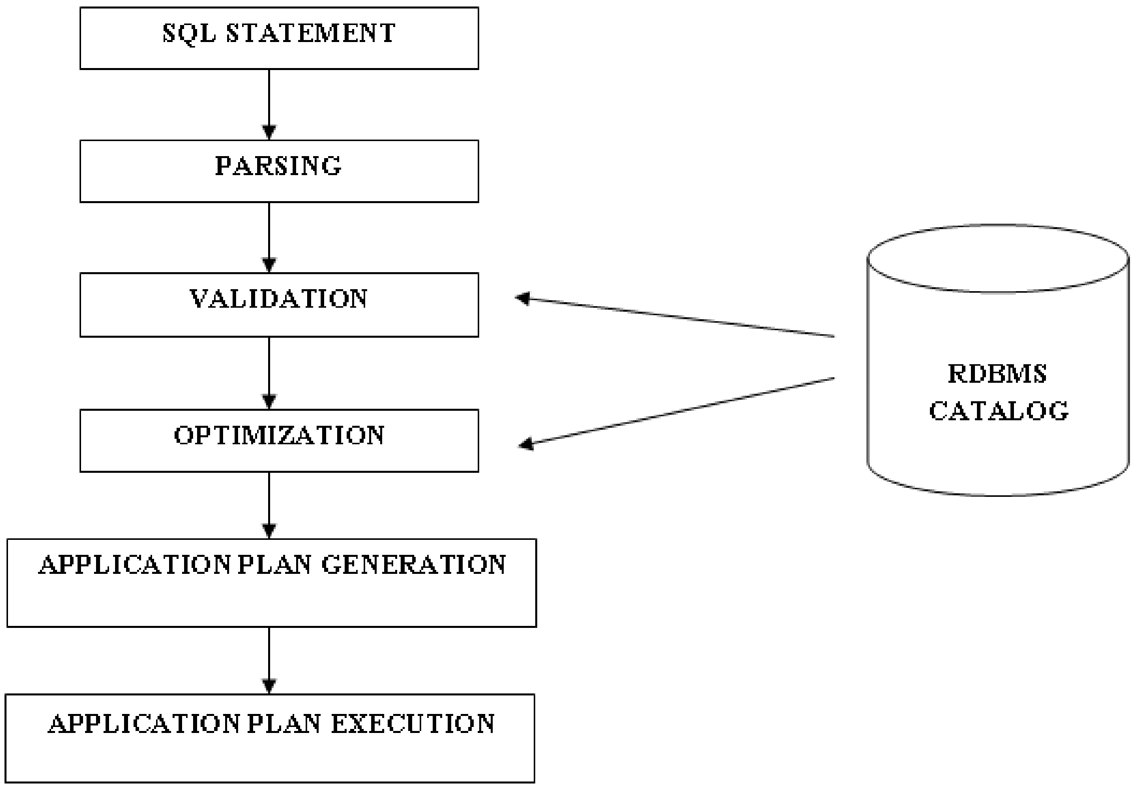

Fig. 1.1 Execution of SQL Statement

Basically, Embedded SQLs are SQL statements included in the programming languages. Any relational databases follow five stages for processing SQL statements – Parsing, Validation, Optimization, Application Plan Generation and Application Plan Execution. The best practice in SQL is to move the maximum number of steps of SQL execution to the program development for faster execution of the Embedded SQL program. These

Embedded SQL features are collectively known as Static SQL. The Static SQL is an adequate and the most efficient way of using SQL in real world application development.

There is a limitation in this process. The method of pre-compilation and application plan generation may not work all situations. There are some general purpose applications such as ad hoc query processing or database design tools, for which it is not possible to provide all the necessary information during compilation prior to runtime. Hence there is need to construct and execute SQL statements 'on the fly' basis. This is possible by using PREPARE and EXECUTE commands in SQL. This type of Embedded SQL programming is known as Dynamic SQL. The basic idea in Dynamic SQL is not to hard code an Embedded SQL statement into the source code, but let the program build the text for the SQL statement during runtime. Then pass the SQL statement text to the RDBMS for execution 'on the fly'. Because these processes happen during runtime, Dynamic SQL is obviously less efficient than Static SQL. For this very reason, the Static SQL statements are widely used whenever possible in database programming.

III. RELATED WORK

In literature survey, among many frameworks and approaches, the possible approaches we observed are probabilistic and fuzzy inferences with machine learning capabilities. However our aim is to propose a methodology which eliminates redundancy by using SQL statements. Data are said to be of high quality if they represent real world constructs. Any data quality research includes accuracy, correctness, completeness and timeliness of data.

Alvaro E Monge proposed a priority queue method which reduces the number of pair-wise comparisons of tuples [3]. Helena Galhardas proposed an execution model which uses declarative language and algorithms, similar to SQL commands to express data cleaning specifications and perform the data cleaning in efficient fashion [4].These proposed models and rules are static and hard to code and manipulate. Vijayshankar Raman describes an interactive data cleaning framework that lets the users see the changes in the data with the help of a spreadsheet like interface [5]. This proposed framework uses gradual construction of transformations through examples using a graphical user interface. But this framework is somewhat rigid and also the detection of anomalies is by human visual inspection is problematic. Presently, several researchers are also encountering similar redundancy problems in the context of the Semantic Web when constructing and merging ontology's [6]. Shahri's proposed a novel duplicate elimination framework which uses fuzzy inference along with the machine learning capabilities [7].This framework lets the users clean the data flexibly and efficiently without requiring any coding. Hard coding for elimination of redundancy based on a relational schema is still tedious and time consuming.

Data Cleaning is an inevitable universal problem when combining data from various distributed and commercial operational databases. Because there is no unified standard spans to all the distributed data sources. Ajeet A. Chikkamannur proposed a straight forward SQL commands access to manipulate with relations [8]. This helps in improving the SQL towards naturalness. Tuple matching is a similar to duplicate detection. Even with careful integration of databases, redundancy may occur due to disagreements among several data sources. Periklis Andritsos used probabilistic approach to clean the dirty databases [9]. This query rewriting technique assigns true probabilities to tuples. These methods showed are both scalable and efficient. Saumya Goyal discussed the overview of hybrid databases along with SQL and NoSQL usages [10]. This theoretical framework is an approach to bridge the gap between SQL and NoSQL in order to query the unstructured data.

IV. METHODOLOGY

In this section, we present our methodology steps to remove the data redundancy via using SQL.

Step 1 : Create new_table using original_table –

    It is to create a new table just by changing the name of the existing table.

Step 2 : Insert into new_table distinct records –

    Insert into new_table values select (distinct column (key), columns,......) from original_table;

Step 3: Drop the records of original_table –

    Delete * from original_table;

    In this step, we can drop the original table contents and also can create SQL code to clear indexes and incremental values in the table.

Step 4: Insert record on the new_table back to the original_table –

    Insert into original_table values select columns from new_table;

Step 5: Drop the new_table –

    Drop table new_table;

In Step 1, the new table is created using an existing table. It is very easy to create a new table using original table just by changing the name. The user has to give the new table name and no need to alter the schema. In Step 2, distinct keyword is used in the select clause to retrieve only distinct tuples in the original table to the new table. This results in duplication elimination. Here the new table consists of tuples without any redundancy. The original table contents are removed in Step 3. Then in Step 4, the new table values are inserted into original table which contains no duplicates. Hence the redundancy has been eliminated via SQL query statements. In Step 5, the newly created new_table, which is no longer needed, has been dropped. User can make use of original table contents which contains no duplicates.

V. EXPERIMENTAL EVALUATION

We have created a practical methodology of the above described theoretical approach for the elimination of redundancy in databases. This methodology is implemented using combination of Static SQL and Dynamic SQL statements. These database programming languages helps us to determine plausible ways to implement elimination of duplicates in the database. These can be easily embedded into any application programs which support database approaches.

The following statement shows usage of GROUP BY, HAVING, ORDER BY in one query, and returns the results with redundant column along with its count. This would be helpful in dealing with large database tables with high redundancy.

```
SELECT Dupl_Column,
         COUNT(*) Total
```

```
FROM   original_table
GROUP  BY Dupl_Column
HAVING COUNT(*) > 1
ORDER  BY COUNT(*) DESC;
```

In order to eliminate more number of duplicates in the database table, the size of the ID set can be used in where clause. The limit can be added with required number within the inner query in order to scan and retrieve the maximum number of duplicates.

VI. CONCLUSION

We have introduced an ameliorated methodology to eliminate redundancy in databases with the help of SQL. We presented a methodology that has been written in SQL queries posed over large tables with duplicate data. Our experimental analysis showed that the resultant data with no duplicates. All the duplicates are eliminated and it also gives the count of duplicates in the database table. This ameliorated methodology is both intuitive and scalable. The methodology can be applied to large database tables with more duplicates.

This work in SQL however opens several views for research in future. We would like to extend this work to eliminate spurious and dangling tuples in the relational and commercial databases. In this way, SQL can be ameliorated towards naturalness.

VII. Acknowledgements

I would like to thank my guide, Professor Dr. Ajeet A. Chikkamannur for his valuable support and sharing knowledge.

REFERENCES

[1] Colin Ritchie, *Relational Database Principles*, 2nd edition, Thomson Learning, 2004.

[2] Elmasri, Navathe, *Fundamentals of Database Systems*, 5th Edition, Pearson Education, 2008.

[3] A.E. Monge and P.C. Elkan, *An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records*, Proc. Sigmod 1997 Workshop on Data Mining and Knowledge Discovery, 1997, pp. 23–29.

[4] H. Galhardas et al., *Declarative Data Cleaning: Language, Model and Algorithms*, Proc. 27th Int'l Conf. Very Large Databases, 2001, pp. 371–380.

[5] V. Raman and J.M. Hellerstein, *Potter's Wheel: An Interactive Data Cleaning System*, Proc. 27th Int'l Conf. Very Large Databases, 2001, pp. 381–390.

[6] R. Guha, *Semantic Negotiation: Co-identifying Objects Across Data Sources*, AAAI Spring Symp. Semantic Web Services Workshop, 2004.

[7] H. H. Shahri, S.H. Shahri, *Eliminating Duplicates in Information Integration: An Adaptive, Extensible framework*, IEEE Computer Society, 2006, pp. 63-71.

[8] Ajeet A. Chikkamannur, Dr. Shivanand M. Handigund, *A Concact Semiotic for recursion in SQL*, International Journal of Emerging trends & Technology in Computer Science (IJETTCS), Vol. 2, pp. 204-210, 2013.

[9] P. Andritsos, A. Fuxman, R.J. Miller, *Clean Answers over Dirty databases: A probabilistic Approach*, 22nd Int'l Conf. on Data Engineering, IEEE, 2006.

[10] S. Goyal, P.P. Srivastava, Anil Kumar, *An Overview of Hybrid Databases*, Int'l Conf. on Green Computing and Internet of Things, IEEE, 2015, pp. 285-288.